

# Fractional programming formulation for the vertex coloring problem

Tomomi Matsui\*, Noriyoshi Sukegawa† and Atsushi Miyauchi‡

*Graduate School of Decision Science and Technology, Tokyo Institute of Technology, Ookayama 2-12-1,  
Meguro-ku, Tokyo 152-8552, Japan*

## Abstract

We devise a new formulation for the vertex coloring problem. Different from other formulations, decision variables are associated with the pairs of vertices. Consequently, colors will be distinguishable. Although the objective function is fractional, it can be replaced by a piece-wise linear convex function. Numerical experiments show that our formulation has significantly good performance for dense graphs.

## 1 Introduction

The vertex coloring problem (VCP) is a well-known NP-hard [4] combinatorial optimization problem with a large number of applications including scheduling, register allocation, and timetabling (see the survey [9] for the details). In this problem, we are given a simple and undirected graph  $G = (V, E)$ . The objective is to find an assignment of colors to  $V$  such that no two adjacent vertices share the same color and the number of colors used is minimized.

In the standard formulation for VCP, letting  $C$  be a set of colors, we introduce a decision variable  $x_{vc}$  ( $\forall v \in V, \forall c \in C$ ) which takes 1 if  $v$  receives color  $c$  and takes 0 otherwise. Since every graph can be colored with  $n = |V|$  colors, it suffices to set  $C = \{1, 2, \dots, n\}$ . Although this formulation is intuitive and simple, there exists a strong symmetry in the feasible region resulting from the indistinguishability of colors. Suppose that we have a solution using  $k$  colors. Then we see that this model has  $\binom{|C|}{k} k!$  equivalent solutions. This property will be a great disadvantage when we use ILP solvers. For this reason, cuts that remove the symmetry have been studied [11, 12]. On the other hand, recently, alternative formulations for VCP have received a considerable attention. For instance, there are studies on a set partitioning formulation [10], an asymmetric representative formulation [2, 3], an unconstrained quadratic binary programming formulation [8], and a supernodal formulation [1]. For further discussion on these formulations, see Burke et al. [1].

In this study, we focus on the pairs of vertices which can be colored by the same color, and associate decision variables with these pairs. As a result, we obtain a new formulation for VCP. Our model does not suffer the symmetry which is discussed above and has a linear fractional objective function. This objective function can be equivalently replaced by a piece-wise linear convex function, which gives us a mixed integer linear programming (MILP) model for VCP. By this transformation, we can feed our model to commercial MILP solvers such as Gurobi Optimizer. To verify the validity of our formulation, we conducted numerical experiments on random graphs and several instances from DIMACS Implementation Challenge, and confirmed that it has a significantly good performance for dense graphs. It should be noted that high edge density does not necessarily make instances easy. In fact, there is a dense but hard instance DSJC125.9 with only 125 vertices from DIMACS Implementation Challenge. The optimal value of this instance was an open problem until very recently. See Gualandi and Malucelli [6] for the details. We confirmed that our model solves this instance less than only 1 minute.

---

\*e-mail: matsui.t.af@m.titech.ac.jp

†e-mail: sukegawa.n.aa@m.titech.ac.jp

‡e-mail: miyauchi.a.aa@m.titech.ac.jp

## 2 Our formulation

### 2.1 Expression as a fractional programming problem

In our formulation, for each distinct pair of vertices  $u$  and  $v$ , we introduce a decision variable  $x_{uv}$  which takes 1 if  $u$  and  $v$  share the same color and takes 0 otherwise. Clearly, we have  $x_{uv} = 0$  for each  $\{u, v\} \in E$ . Here, we use the following inequality constraints

$$x_{uv} + x_{vw} - x_{uw} \leq 1 \quad (\forall u, v, w \in V \text{ with } u \neq v, v \neq w, u \neq w)$$

to obtain an explicit description of the feasible region. These inequalities say that if  $u$  and  $v$  share the same color ( $x_{uv} = 1$ ) and  $v$  and  $w$  also share the same color ( $x_{vw} = 1$ ), then  $u$  and  $w$  must receive the same color ( $x_{uw} = 1$ ). These inequalities are referred to as the triangle inequalities studied in Grötschel and Wakabayashi [5] as facet-defining inequalities for a clique partitioning polytope. This relationship is natural because if  $\mathbf{x}$  is a feasible solution for VCP, then a set  $E_{\mathbf{x}} = \{\{u, v\} \mid x_{uv} = 1\}$  of edges induces a clique partitioning of the complement graph  $\overline{G}$  of  $G$ , and vice versa.

Next, let us consider how to calculate the number of colors used in  $\mathbf{x}$ , namely the objective value. To this aim, we focus on the number of connected components in  $(V, E_{\mathbf{x}})$ . It is easy to see that this number equals the desired value. For a feasible solution  $\mathbf{x}$ , let us define

$$f_v(\mathbf{x}) = \frac{1}{1 + \sum_{u \in V} x_{uv}}$$

for each  $v \in V$ . Suppose that a vertex  $v$  belongs to a connected component  $(V', E')$  with  $|V'| = k$  in  $(V, E_{\mathbf{x}})$ . Then we have  $f_v(\mathbf{x}) = 1/k$  since  $V'$  is a clique of  $(V, E_{\mathbf{x}})$ . Thus, the sum of  $f_v(\mathbf{x})$  over  $v \in V'$  equals 1 for each connected component  $(V', E')$ , which implies that the sum of  $f_v(\mathbf{x})$  over  $v \in V$  gives the number of connected components in  $(V, E_{\mathbf{x}})$ . Therefore, we obtain the following proposition.

**Proposition 1.** *For a given feasible solution  $\mathbf{x}$ ,*

$$\sum_{v \in V} f_v(\mathbf{x})$$

*equals the number of connected components in  $(V, E_{\mathbf{x}})$ , which is the number of colors used in  $\mathbf{x}$ .*

In sum, our formulation is described as follows:

$$\begin{aligned} & \text{minimize} && \sum_{v \in V} f_v(\mathbf{x}) \\ & \text{subject to} && x_{uv} = 0 && (\forall \{u, v\} \in E), \\ & && x_{uv} + x_{vw} - x_{uw} \leq 1 && (\forall u, v, w \in V \text{ with } u \neq v, v \neq w, u \neq w), \\ & && x_{uv} \in \{0, 1\} && (\forall u, v \in V \text{ with } u \neq v). \end{aligned}$$

It should be noted that there are redundant variables and constraints. Suppose that  $\{u, v\} \in E$ . Then, of course, we do not need to prepare the decision variable  $x_{uv}$ . In addition, the transitivity constraints  $x_{uv} + x_{vw} - x_{uw} \leq 1$  is redundant for every  $w \in V \setminus \{u, v\}$  because it is equivalent to  $x_{vw} - x_{uw} \leq 1$ , which is satisfied by any pair of  $x_{vw}$  and  $x_{uw}$  with  $0 \leq x_{vw}, x_{uw} \leq 1$ . In our numerical experiments, such redundant variables and constraints are removed.

### 2.2 Expression as a mixed integer linear programming problem

When implementing our formulation on MILP solvers, we substitute a piece-wise linear convex function for the fractional objective function. For each  $v \in V$ , we introduce a continuous decision variable  $f_v$

Table 1: Results for the randomly generated graphs

Instance		Our formulation	
$n$	$p$	Time[s]	Gap[%]
30	0.3	3.44	—
	0.5	4.48	—
	0.7	0.17	—
	0.9	0.04	—
50	0.3	*****	16.43
	0.5	**	10.06
	0.7	11.71	—
	0.9	0.14	—
70	0.3	*****	33.00
	0.5	*****	17.00
	0.7	517.87	—
	0.9	0.40	—

Table 2: Results for the randomly generated graphs

Instance		Our formulation	
$n$	$p$	Time[s]	Gap[%]
100	0.9	1.50	—
150	0.9	61.59	—
200	0.9	*	1.59

which equals  $f_v(\mathbf{x})$  for a given feasible solution  $\mathbf{x}$ . Namely, the objective function will be the sum of  $f_v$  over  $v \in V$ . For each  $v \in V$  and for each  $i \in \{0, 1, \dots, n-1\}$ , we add the following linear inequality constraint

$$f_v \geq u_i \left( \sum_{u \in V} x_{uv} \right),$$

where

$$u_i(d) = -\frac{1}{(i+1)(i+2)}(d-i) + \frac{1}{i+1}.$$

If  $d \in \{0, 1, \dots, n-1\}$ , then  $u_d(d) = 1/(1+d)$  and  $u_d(d)$  is the largest value among  $\{u_k(d) \mid k \in \{0, 1, \dots, n-1\}\}$ . Hence, if  $\mathbf{x}$  is a feasible solution, since  $d := \sum_{u \in V} x_{uv}$  is an integer and  $f_v$  is minimized in the objective function, we have

$$f_v = u_d(d) = \frac{1}{1+d} = \frac{1}{1 + \sum_{u \in V} x_{uv}} = f_v(\mathbf{x})$$

for every  $v \in V$ . This shows the validity of the above constraints.

### 3 Numerical experiments

In this section, we report the numerical experiments on our formulation. All results were measured on a Linux-based computer with 2.66 GHz quad-core processors and 24 GB RAM. We used

Table 3: Results for middle-sized dense instances from the Second DIMACS Implementation Challenge

Instance		Density[%]	Time[s]	Best Bounds		Opt.
Name	$n$			Lower	Upper	
r125.1c	125	96.8	0.75	46	46	46
DSJC125.9	125	89.8	57.48	44	44	44 [6]
DSJC250.9	250	89.6	7200	71	72	72 [7]
DSJR500.1c	500	97.2	7200	79	86	85 [6]

Gurobi Optimizer 5.6.0 to solve the MILP problems, and imposed a time limit of two hours (7200s) on run time per instance. In addition, below, “gap” means the relative gap, i.e., (upper bound – lower bound)/upper bound.

The results on random graphs are shown in Table 1. We generate five instances for each pair of the edge density  $p \in \{0.3, 0.5, 0.7, 0.9\}$  and the number of vertices  $n \in \{30, 50, 70\}$  and exhibit the average computation time for these five instances. When calculating the average, if there is an instance which cannot be solved to optimality by the limit, (since the average does not make sense) we count the number of such instances and denote this by the number of “\*”s. We see that, unfortunately, our formulation fails to solve several instances with  $p \leq 0.5$  even when  $n = 50$ . However, for dense graphs, especially when  $p = 0.9$ , the computation time is very short.

Motivated by the results in Table 1, we further solved three dense and a little bit larger instances. These three instances are again randomly generated instances where the edge density  $p$  is fixed to 0.9 and the number of vertices  $n$  ranges in  $\{100, 150, 200\}$ . Again, for each  $n$ , we generate five instances. We see that, even when  $n = 150$ , optimal solutions are obtained within only about 1 minute, on average. Although, when  $n = 200$ , one instance could not be solved to optimality, the remained relative gap is small.

Next, in Table 3, we show results for several dense and middle-sized instances from the Second DIMACS Implementation Challenge. The three instances DSJC125.9, DSJC250.9, and DSJR500.1c are not so large but very hard. Indeed, their optimal values were unknown until very recently. See [6] for DSJC125.9 and DSJR500.1c, and [7] for DSJC250.9. Although we could not solve the last two instances within two hours, the first two instances are solved to optimality. Moreover, its computation time is short.

## 4 Conclusion

In this study, we devise a new formulation for VCP. Different from other existing formulations, we associate decision variables with the pairs of vertices, which makes the colors distinguishable. In our formulation, the number of colors used is expressed as a linear fractional function. To implement the formulation on MILP solvers, we replace the fractional objective function by a piece-wise linear convex function. The noteworthy property of our formulation is its considerably high performance for dense graphs.

## References

- [1] E. K. Burke, J. Mareček, A. J. Parkes, and H. Rudová. A supernodal formulation of vertex colouring with applications in course timetabling. *Ann. Oper. Res.*, 179(1):105–130, 2010.

- [2] M. Campêlo, V. A. Campos, and R. C. Corrêa. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Appl. Math.*, 156(7):1097–1111, 2008.
- [3] M. Campêlo, R. Corrêa, and Y. Frota. Cliques, holes and the vertex coloring polytope. *Inf. Process. Lett.*, 89(4):159–164, 2004.
- [4] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. *WH Freeman & Co., New York*, 1979.
- [5] M. Grötschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Math. Program.*, 47(1-3):367–387, 1990.
- [6] S. Gualandi and F. Malucelli. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS J. Comput.*, 24(1):81–100, 2012.
- [7] S. Held, W. Cook, and E. C. Sewell. Maximum-weight stable sets and safe lower bounds for graph coloring. *Math. Program. Comput.*, 4(4):363–381, 2012.
- [8] G. A. Kochenberger, F. Glover, B. Alidaee, and C. Rego. An unconstrained quadratic binary programming approach to the vertex coloring problem. *Ann. Oper. Res.*, 139(1):229–241, 2005.
- [9] E. Malaguti and P. Toth. A survey on vertex coloring problems. *Int. Trans. Oper. Res.*, 17(1):1–34, 2010.
- [10] A. Mehrotra and M. A. Trick. A column generation approach for graph coloring. *INFORMS J. Comput.*, 8(4):344–354, 1996.
- [11] I. Méndez-Díaz and P. Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Appl. Math.*, 154(5):826–847, 2006.
- [12] I. Méndez-Díaz and P. Zabala. A cutting plane algorithm for graph coloring. *Discrete Appl. Math.*, 156(2):159–179, 2008.

## Adding simple cuts

In this appendix, we introduce simple valid inequalities to strengthen the formulation. Let  $|I_v|$  be the size of a maximum independent set  $I_v$  which includes  $v \in V$  in  $G$ . Then, it is easy to see that the number of vertices which can receive the same color as that of  $v$  is bounded above by  $(|I_v| - 1)$  in any feasible solution. In other words, for each  $v \in V$ , the following inequality

$$\sum_{u \in V} x_{uv} \leq |I_v| - 1$$

can be added to our formulation. Table 4 and Table 5 show the results of our formulation with these simple cuts for the instances shown in Table 2 and Table 3, respectively. The computation time includes the preprocessing time, namely the computation time for calculating  $|I_v|$  for each  $v \in V$ . This preprocessing time is exhibited in the brackets. Although we also conducted numerical experiments on the instances shown in Table 1, the improvement on solving time is not significant, hence, we omit the corresponding results. From Table 4 and Table 5, we see that simple cuts provide an advantage on solving time for all instances other than DSJC250.9. For DSJC250.9, the upper bound becomes slightly worse.

Table 4: Results for the randomly generated graphs with simple cuts

Instance		With simple cuts	
$n$	$p$	Time[s]	Gap[%]
100	0.9	1.26 (0.22)	—
150	0.9	59.97 (0.67)	—
200	0.9	1587.26 (1.54)	—

Table 5: Results for middle-sized dense instances from the Second DIMACS Implementation Challenge with simple cuts

Instance			Time[s]	Best Bounds	
Name	$n$	Density[%]		Lower	Upper
<b>r125.1c</b>	125	96.8	0.27 (0.09)	46	46
<b>DSJC125.9</b>	125	89.8	33.63 (0.40)	44	44
<b>DSJC250.9</b>	250	89.6	7200 (3.50)	71	73
<b>DSJR500.1c</b>	500	97.2	7200 (1.86)	79	85